AUTOMATED PRICING FOR INDUSTRIAL GOODS USING FUZZY LOGIC

Mario Thron[1], Marian Witte[2], Giuliano Persico[2]

[1]Institut f. Automation und Kommunikation e.V., Werner-Heisenberg-Str. 1, 39106 Magdeburg, Germany

[2]Demag Cranes & Components GmbH, Ruhrstraße 28, 58300 Wetter, Germany

Abstract

Pricing complex industrial products often involves manual assessment of dynamic factors such as factory load, regional purchasing power, and customer success rates. This paper proposes an automated solution for the final pricing stage, integrating manufacturer and customer-specific conditions using a Service-Oriented Architecture (SOA). Real-time data from ERP systems and MES is processed via web services, while fuzzy logic formalizes expert knowledge into pricing rules. Sales engineers input basic case characteristics, and the system dynamically computes the price factor. An open-source implementation of IEC 61131-7 for fuzzy control programs demonstrates feasibility, showing that automation enhances decision-making, supports knowledge transfer, and offers flexibility for future integrations.

Keywords: *Sales Management, Fuzzy Logic, Pricing, Configurable Industrial Goods*

1. INTRODUCTION

Selling complex goods in the industrial sector often requires individualized configurations and close interaction between customers and the manufacturer's sales engineers as reported by the Dock Team (2025). Unlike standard consumer products, these goods cannot be priced using static catalog values alone. Instead, the pricing process is a multi-stage task, typically divided into two key phases. First, manufacturers define "master prices" for each product or option. These are essentially the base prices listed in internal catalogs and are based on predictions, such as those of material, energy, and labor costs. Second, these base prices are adjusted to reflect manufacturer- and customer-specific conditions, resulting in the final price offered to the customer.

The second stage of pricing, which involves adjusting the master price according to dynamic and situational factors, poses significant challenges. Pricing decisions must consider subjective factors such as the current factory load, the customer's regional economic context, the success rates of past negotiations, and the strategic interests of the manufacturing company in certain regions or application domains. Consequently, this process is often handled manually, which makes it time-consuming, inconsistent, and difficult to scale or standardize.

In response to these challenges, this paper introduces an automated approach to computing pricing factors in the final pricing stage. Leveraging a Service-Oriented Architecture (SOA) and integrating data from internal systems, such as ERP and MES, as well as external sources, the system enables real-time, rule-based pricing decisions. Expert knowledge is formalized into fuzzy logic rules as invented by Zadeh, L.A. (1965), enabling nuanced, human-like reasoning in the price adjustment process. Sales engineers only need to input the core case characteristics, such as customer region, delivery date, or reference number, to receive a dynamically calculated pricing factor to adjust the master price.

This work aims to demonstrate that automation in the pricing process can improve consistency, transparency, and decision quality, while also facilitating knowledge transfer within sales organizations. The proposed approach has been validated through an open-source implementation and a web-based demonstrator, showing that such a system can be both technically feasible and organizationally beneficial.

In the InnoSale research project, which is part of the EUREKA ITEA cluster, full product implementation was not the intended goal. The prototype was deliberately designed to validate

feasibility rather than perfection (InnoSale Project 2025). The prototype comprises functional services, a fuzzy-logic pricing engine, and an editor to demonstrate core capabilities. The limited integration work reveals manual handovers and missing interfaces that will require seamless automation in subsequent phases.

2. MATERIALS AND METHODS

This section outlines the technical and procedural basis for the proposed approach to automated pricing factor calculation for industrial goods as defined by the InnoSale Project (2025). It covers the system architecture, data flow, rule formalization process, and the logic evaluation method used, ensuring a reproducible and modular implementation strategy.

2.1. System Architecture and Data Flow

The Pricing Platform is built on a three-tier, service-oriented architecture (SOA) with a centralized Identity and Access Management (IAM) framework enforcing authentication, authorization, and audit logging across every component. **Figure *1*** illustrates the overall structure and data flow.

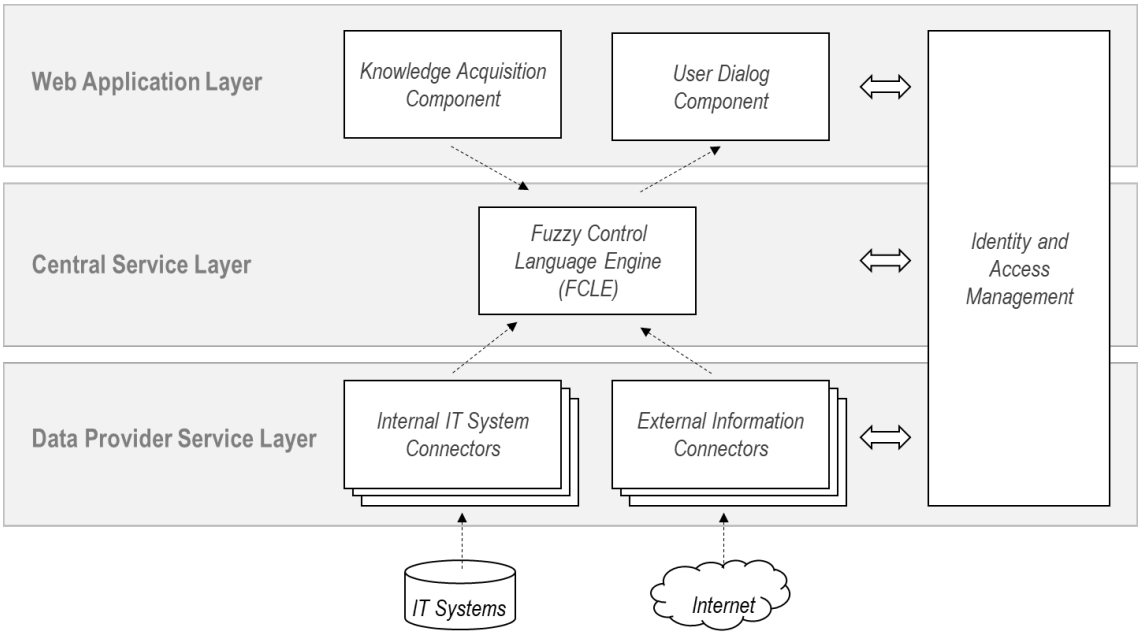

Figure 1: General Information Flow in the SOA-based Pricing Platform.

In the *Data Provider Service Layer*, a collection of internal and external connectors continuously harvests and normalizes the inputs required for pricing computations. Internally, adapters link to on-premises systems, such as ERP, MES, CRM, and PLM, pulling factory utilization, order backlogs, inventory status, and resource availability as REST/SOAP messages, JDBC queries, or event feeds. Externally, public APIs and data feeds supply macroeconomic indicators, such as purchasing power parity and market-risk indices, as well as commodity prices and industry benchmarks. Each connector authenticates using a service account credential issued by the IAM module.

The Fuzzy Control Language Engine (FCLE) is at the heart of the *Central Service Layer*. It applies rule-based inference to raw inputs. Sales case parameters and operational metrics flow into the FCLE as authenticated, JWT-secured API calls. The engine then evaluates the membership functions and fuzzy-logic rules stored in a version-controlled repository and computes the pricing factors, either on demand or in queued batches. Caching ensures low latency.

The *Web Application Layer* provides two primary user interfaces. The User Dialog Component is a single-page application through which sales engineers enter customer reference numbers, target delivery dates, geographic regions, and product configuration codes. The component then invokes FCLE endpoints to retrieve real-time pricing factors and present them in a pricing dashboard. The Knowledge Acquisition Component provides an administrative console for creating, importing, reviewing, approving, and publishing new or updated fuzzy-logic pricing rules, with updates to the FCLE rule set.

The *Identity and Access Management (IAM)* module underlies all layers and offers single sign-on (SSO) via OAuth 2.0/OpenID Connect, role-based access control (RBAC) for fine-grained UI and API permissions, optional multi-factor authentication (MFA), and centralized session management. The module issues and validates JSON Web Tokens for user sessions and inter-service calls and retains a complete audit log of authentication events, authorization decisions, and data access operations. The IAM framework unifies user and service identities to enforce a consistent security posture, simplify compliance reporting, and streamline the onboarding of new systems and components across the platform.

With the architecture and security concept in place, we now detail each core component: the Data-Providing Services for input aggregation; the Fuzzy Logic Editor and Engine for rule authoring and inference; the Pricing UI for sales-engineer interaction; the Knowledge Acquisition console for rule management; and the IAM module for unified, secure access control.

2.2 Data-Providing Services

Automated pricing relies on accurate and timely input. Data-providing services decouple each information source from the pricing engine, ensuring that data can be added, updated, or replaced independently without disrupting the rest of the system. The platform maintains flexibility by encapsulating each source behind a well-defined API or connector. This allows new metrics to be integrated, legacy systems to be modernized, and failures in one service to not cascade into the pricing computation.

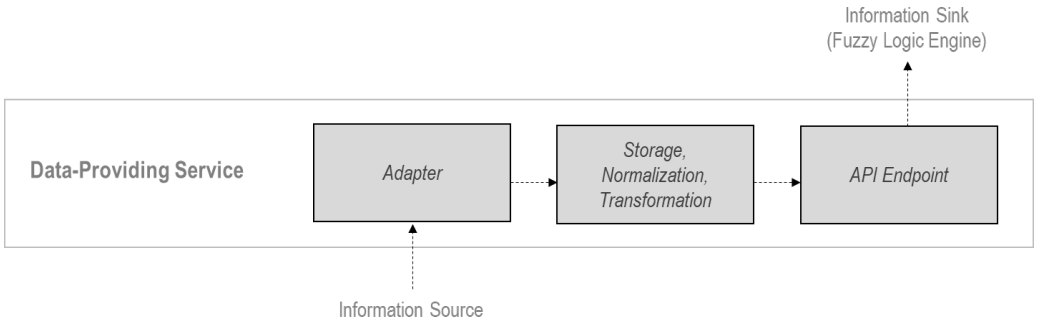

Figure 2: Structure of a Data-Providing Service

As **Figure *2*** depicts, typical data-providing service connectors consist of three layers:

1. Adapter: Implements the protocol (e.g., REST, SOAP, JDBC, file watch) required to extract raw data from an ERP, MES, CRM, or an external feed.
2. Storage, Normalization, and Transformation: Converts source-specific formats into a common schema (for example, mapping "factory_load" values to a 0–100 utilization percentage) and applies any necessary storage, prediction, smoothing or filtering logic.
3. API Endpoint: Exposes the normalized data via a secure, authenticated interface (e.g., a RESTful JSON endpoint secured with OAuth 2.0), allowing the fuzzy-logic engine to poll or subscribe to changes.

For example, the Big Mac Index Provider developed in the InnoSale project is a REST-based FastAPI service that computes and supplies currency-valuation metrics based on the Big Mac Index, an economic

indicator comparing the purchasing power of different currencies via locally observed Big Mac prices. Internally, it cleanly maps onto our three data-providing service layers:

- *Adapter*: Upon startup, the service automatically fetches raw price and exchange rate data from publicly available datasets sourced from *The Economist* and its official Git repository via HTTP. The service supports on-demand queries for current valuations, multi-year historical analyses, and trend projections for future months.
- *Storage, Normalization & Transformation*: Raw inputs—price and currency pairs specified by ISO codes—are stored and converted into standardized valuation scores that quantify deviations from purchasing-power parity between a producer currency (e.g., USD) and a quote currency (e.g., EUR). Other currency-pair conversions can be accommodated just as easily.
- *API Endpoint*: Three authenticated REST endpoints are exposed under a single HTTP host:
  - */bigmacindex:* returns the latest valuation for any specified currency pair,
  - */bigmacindexhistory:* delivers a historical series over a user-selectable number of years,
  - */bigmacindextrend:* produces a forecast of future index values for a specified number of months.

All endpoints are documented via an auto-generated OpenAPI interface (e.g., at "*http://<host>:<port>/docs*"), enabling both human testing (see **Figure *3***) and seamless machine integration. These endpoints feed directly into the fuzzy-logic engine to adjust pricing factors in real time.

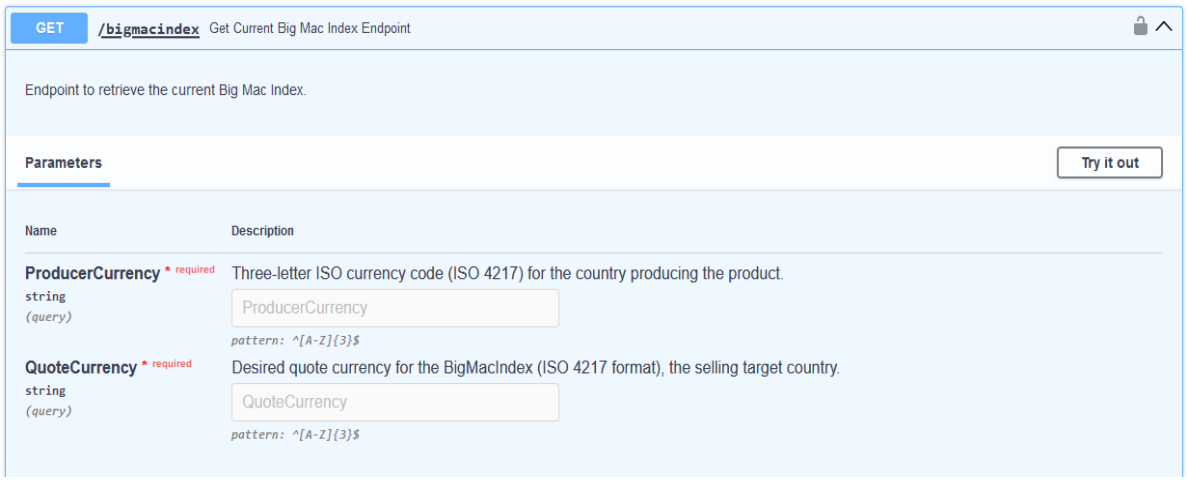

Figure 3: Big Mac Index Provider – Section of the User Interface for Testing the Service

2.3 Fuzzy Control Language Engine (FCLE)

At the heart of the Central Service Layer sits the Fuzzy Control Language Engine (FCLE), which acts as a dedicated reasoning service. It transforms raw inputs, such as factory utilization, market indicators, and production costs, into a cohesive pricing factor. FCLE is a standalone microservice within the InnoSale SOA that receives input variable values via REST calls. It applies the knowledge of human experts, such as CEOs, heads of sales, and advanced sales engineers, which is formalized in the IEC 61131-7 function blocks, as defined by the International Electrotechnical Commission (2000). It then returns precise adjustments that determine the final price.

At startup, the FCLE parser ingests a Base64-encoded FCL file and interprets its FUZZIFY, DEFUZZIFY, and RULEBLOCK sections to construct an in-memory fuzzy function block. Each input variable is associated with multiple point term definitions, which are piecewise-linear membership functions, while output variable singletons are initialized with their defuzzification method (center-of-gravity for singletons, or COGS) and default fallback values. When a pricing request arrives at the

evaluate endpoint, the engine resets the previous state. It then fuzzifies every input by interpolating membership degrees and evaluates each rule by combining its antecedent memberships (e.g., via MIN) and accumulating conclusions (e.g., via MAX). Finally, defuzzification computes a single numeric factor guaranteed to lie within the bounds implied by the singleton terms, ensuring smooth, human-like gradations in price adjustments.

Although FCLE's core logic resides in a single Python file, its modular design cleanly separates concerns. A parser module built on Lark translates FCL text into Python objects. An operation module supplies standard fuzzy operators, such as MAX, MIN, and ASUM. An I/O connectors module handles any necessary lookups of missing variables from a cached store. A service module exposes the FastAPI application with endpoints for evaluation, rulebase management, and configuration management. This layering means that updating pricing heuristics – whether refining a membership curve or tweaking a rule weight – requires only uploading a new FCL file to the "*/set_rulebase*" endpoint, after which the next "*/evaluate*" call will reflect the change.

For example, a simplified pricing block might look like this:

```
FUNCTION_BLOCK Pricing_FB
 VAR_INPUT
  factory_load : REAL;
  market_demand: REAL;
 END_VAR
 VAR_OUTPUT
  pricing_factor: REAL;
 END_VAR
 FUZZIFY factory_load
  TERM low  := (0,1) (50,0);
  TERM high := (50,0) (100,1);
 END_FUZZIFY
 DEFUZZIFY pricing_factor
  TERM decrease := 0.8;
  TERM normal   := 1.0;
  TERM increase := 1.2;
  METHOD : COGS;
  DEFAULT := 1.0;
 END_DEFUZZIFY
 RULEBLOCK AdjustPrice
  AND : MIN;
  ACCU: MAX;
  RULE 1: IF factory_load IS low THEN pricing_factor IS increase;
  RULE 2: IF factory_load IS high THEN pricing_factor IS decrease;
 END_RULEBLOCK
END_FUNCTION_BLOCK
```

The FCLE service provides the following endpoints:

- *POST /evaluate*: Evaluates the Fuzzy Logic for the pricing factor. The request would be JSON encoded information like this: *{ "input_values": { "factory_load":75, "market_demand":60 } }*, and the response would look similar to that *{ "pricing_factor":1.03 }*.
- *POST /set_rulebase*: Uploads a Base64-encoded FCL file. Workflow tools call this endpoint to publish new or revised pricing logic.
- *GET /get_rulebase*: Retrieves the active rulebase in Base64-encoded form, enabling administrative review or archival.
- *POST /set_service_conf* and *GET /get_service_conf*: Manage the service's own configuration (e.g., caching parameters, connector settings) via Base64-encoded YAML payloads.

2.4 User Dialog Component

The User Dialog Component is positioned in the Web Application Layer of the project's SOA and works as the interactive core of the pricing workflow. Sales engineers begin by selecting "Price Calculation" from the navigation pane, after which they are presented with a customizable set of input fields reflecting the current pricing variables. In the example shown in **Figure *4***, six factors are presented: monopoly status, delivery time pressure, customer agreement status, buying rate, factory capacity, and stock load. However, this list can be extended or reduced as business needs evolve. The next section will present how to do this.

In our example here, Monopoly Status represents the customer's market share in the relevant segment and is expressed as a percentage; Delivery Time Pressure quantifies the urgency of the requested delivery on a 1–5 scale; Customer Agreement Status indicates whether an existing contract is in place (0 for none, 1 for valid agreement); Buying Rate measures how frequently the customer converts inquiries into orders; Factory Capacity reflects the projected load on production resources; and Stock Load shows the percentage of requested items currently available in inventory.

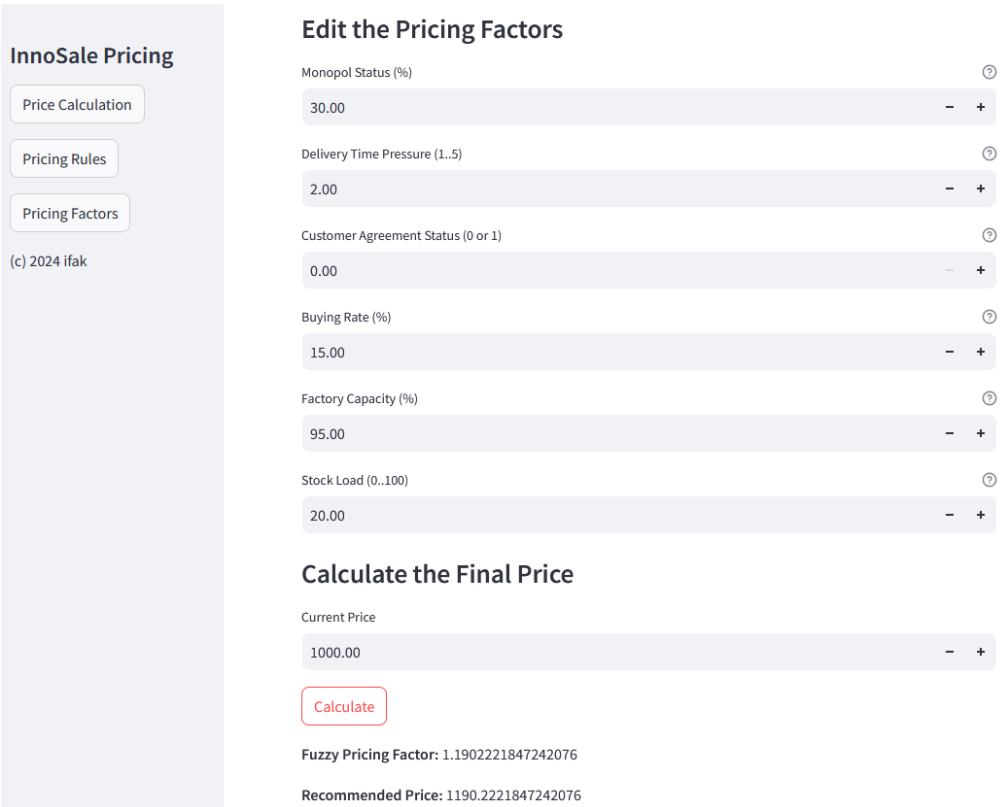

Figure 4: User Dialog Component - The Pricing Interface

Below these operational factors, the "Current Price" field displays the master price – computed from the sum of component prices defined in the internal catalog – and allows any final manual adjustments. When the engineer clicks "Calculate", the interface issues an authenticated call to FCLE's */evaluate* endpoint, passing all entered values. In response, the fuzzy-logic engine returns a Fuzzy Pricing Factor, which captures the net effect of all rules as a single, scalar multiplier. The UI then immediately computes and displays the Recommended Price (the product of the master price and the fuzzy factor), giving engineers clear guidance on the adjusted offer. Sales engineers retain the freedom in managing the price based on their own experiences.

By bringing together configurable inputs, real-time computation, and transparent results in one seamless panel, the User Dialog Component eliminates error-prone spreadsheet work and accelerates decision cycles. Engineers can experiment with "what-if" scenarios, tweaking a delivery pressure or stocking assumption, and instantly observe the impact on both the factor and final price, fostering confidence in the automated pricing process and encouraging its adoption in daily operations.

2.5 Knowledge Acquisition Component

The Knowledge Acquisition Component resides alongside the User Dialog Component in the Web Application Layer. It provides a dedicated administrative console for authoring and managing the fuzzy-logic rule base. Sales engineers rely on the price calculation UI for day-to-day offer adjustments. In contrast, pricing architects and domain experts use this editor to capture, refine, and version the expert knowledge encoded in IEC 61131-7 FCL. The system centralizes rule editing in the browser to ensure that any change to membership functions, rule weights, or defuzzification settings is immediately visible to the entire sales organization and consistently applied by the FCLE engine.

When an administrator selects "Pricing Rules" from the navigation pane, the main panel displays the current function block definition in an embedded code editor. The complete FCL text can be edited directly here. It begins with FUNCTION_BLOCK DynamicPricing and includes VAR_INPUT, FUZZIFY, DEFUZZIFY, and RULEBLOCK sections. The editor supports syntax-preserving copy and paste, allowing experts to import rule snippets from technical specifications or previous projects. Once the revisions are complete, clicking "Upload Rules" a base64-encoded version of the edited text is posted to the FCLE service's */set_rulebase* endpoint. This triggers an immediate reload of the in-memory rule base. Conversely, clicking the "Download Rules" button retrieves the current rule base via the */get_rulebase* endpoint, enabling offline review or archiving.

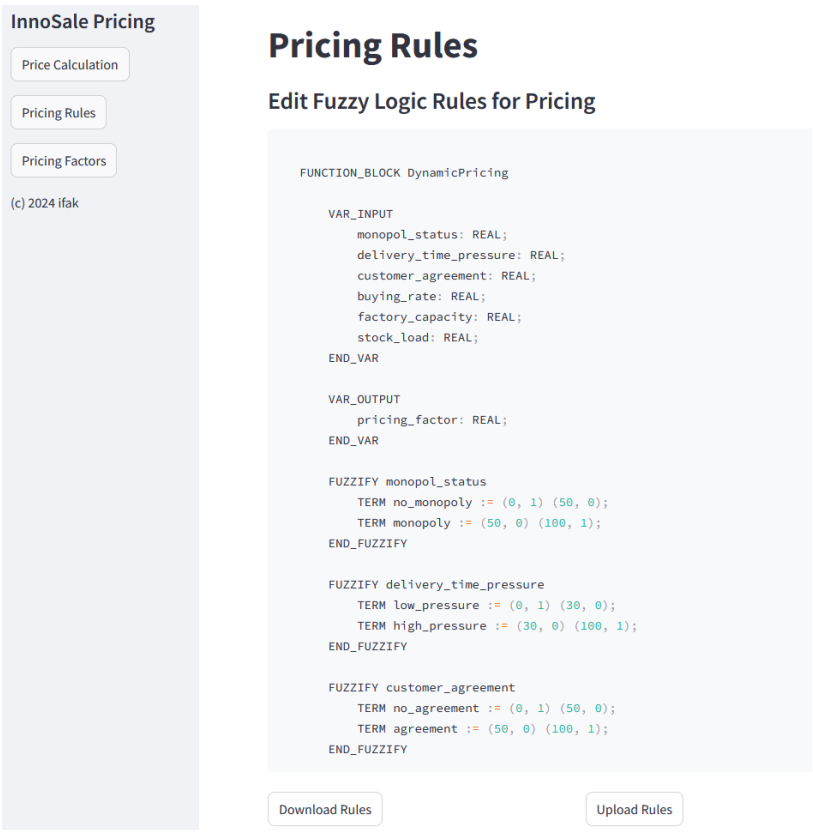

Figure 5: Knowledge Acquisition Component - Edit and deploy fuzzy logic rules

This component provides a live, in-context view of the active rule base, eliminating the need for manual file transfers and scripting overhead. Rule authors can tweak term definitions iteratively, such as by adjusting the (50,0) inflection point for a “high_pressure” term, and insert new conditional clauses. They can then immediately test these changes in sandbox environments before publishing. Thus, the Knowledge Acquisition Component streamlines the maintenance of complex pricing logic and fosters continuous improvement and transparent governance over the fuzzy-logic decision rules that drive automated pricing.

2.6 Identity and Access Management

In any industrial application, ensuring the security and integrity of pricing data is paramount, and in our SOA design, the Identity and Access Management (IAM) module permeates every layer, from data connectors to the FCLE engine and the Web Application Layer. In the browser, the User Dialog and Knowledge Acquisition components both rely on a shared authentication context. Engineers and administrators sign in once via Single Sign-On (SSO) and obtain a JSON Web Token (JWT), which transparently authorizes subsequent calls to REST endpoints. This unified approach spares users from managing multiple credentials and guarantees that every request, whether fetching the current rule base or evaluating a fuzzy pricing factor, carries a verifiable, tamper-proof token.

Behind the scenes, we evaluated Keycloak as our identity provider, leveraging its OpenID Connect support to issue JSON Web Tokens (JWTs) and enforce Role-Based Access Control (RBAC). In production, each microservice validates incoming tokens and extracts roles, such as "sales_engineer" or "pricing_admin," before granting access to protected resources. For experimental or demo deployments, the IAM layer can be disabled entirely to allow open access.

HTTPS secures communication between the client and server, ensuring the confidentiality and integrity of sensitive data in transit. The RESTful services adhere to stateless principles, meaning every request

contains all the necessary information for authentication and authorization with no server-side session state.

Beyond authentication, IAM supports granular permission management. Administrators can assign engineers to groups (e.g., "sales") and bind those groups to roles (e.g., "can_edit_rules"), which allows for fine-grained control over which users can update membership functions, upload new rule bases, or simply view computed pricing factors. All identity data and access events are recorded in a centralized audit log, which simplifies compliance reporting and forensic analysis. Thus, our IAM framework secures the platform and underpins trust in the automated pricing process by making every decision provable and reproducible.

3. RESULTS

The proposed system for automated pricing factor calculation was implemented as a functional prototype and validated through a series of internal demonstrations. This chapter summarizes the key outcomes of the implementation process, with a focus on software functionality, rule evaluation behavior, and system responsiveness.

A fully functional prototype was developed using open-source tools. It integrates a fuzzy logic evaluation engine with a web-based interface and backend service orchestration.

- The fuzzy logic engine was implemented according to the IEC 61131-7 standard for "Fuzzy Control Programming."
- Web services were created for retrieving internal and external data points needed for pricing factor calculations.
- A user interface allows sales engineers to enter case characteristics and receive immediate pricing suggestions.

Two of the services (the Fuzzy Control Language Engine and the Big Mac Index Provider) have been developed in the project and are published via GitHub (ifak-prototypes 2025a, and ifak-prototypes 2025b). They may be used out of the box or be used as blueprints for further own developments.

The system was tested with a representative set of pricing rules and input cases to evaluate the fuzzy logic engine's behavior. The rules were successfully triggered and evaluated based on different combinations of input values. The fuzzy inference mechanism produced smooth, continuous output values for pricing factors, avoiding abrupt jumps typical in threshold-based systems. Defuzzification resulted in pricing factors that aligned with expert expectations for a variety of test scenarios.

Although the demonstrator successfully validated the core fuzzy-logic workflow, its limited scope means that the sales engineer must manually invoke the data-providing services to collect fresh input values and enter them into the user dialog component in the current setup. In a full product implementation, deeper integration between the SOA connectors and the UI would close this gap by automatically fetching and populating real-time data, thereby delivering a seamless, end-to-end pricing experience.

4. DISCUSSION

The development and validation of an automated pricing factor calculation system demonstrate that pricing processes in industrial sales can be effectively supported by a rule-based, service-oriented architecture enhanced with fuzzy logic.

One of the key benefits observed is the system's role in formalizing and preserving domain knowledge. Traditionally, pricing adjustments rely heavily on the tacit judgment of experienced staff, which is difficult to document or transfer. By formalizing pricing logic into fuzzy rules:

- The company can reduce dependency on individual experts.
- Less experienced sales engineers receive support for consistent decision-making.

- Business knowledge becomes auditable and easier to maintain over time.

This approach opens up a pathway toward systematic knowledge engineering in the sales domain.

Despite the promising results, several limitations must be considered:

- *Data Availability and Quality*: The system's accuracy depends on the quality and timeliness of data from internal systems and external APIs.
- *Rule Complexity Management*: As more rules are added, especially those with interacting pricing factors, the rule base may become difficult to manage.
- *Change Management*: Shifting to a partially automated pricing system requires aligning with existing workflows and building trust among experienced sales staff.

The modular architecture allows for easy system expansion. Additional pricing factors, such as political risk, competitor activity, and logistics constraints, can be integrated with minimal changes to the core infrastructure. Further development could include:

- Integration with historical sales databases for automatic rule refinement or training,
- Visualization tools for rule behavior and factor influence analysis,
- Machine learning components to suggest or optimize fuzzy rules over time,
- Manual adjustments of pricing rules in order to get more accurate prices.

5. CONCLUSIONS

This work presented an automated approach for calculating pricing factors in the sales of complex industrial products, targeting the final and often subjective stage of the pricing process. By combining a Service-Oriented Architecture (SOA) with fuzzy logic, the system successfully integrates data from various sources and applies expert knowledge to produce consistent, real-time pricing adjustments. The use of fuzzy logic proved especially effective in formalizing intuitive, experience-based decision-making processes that are otherwise difficult to encode using traditional logic or threshold-based systems.

The prototype implementation demonstrated the technical feasibility of this approach, showing that sales engineers can be effectively supported through a tool that dynamically computes price factors based on basic sales case inputs. Furthermore, the separation of data acquisition, rule definition, and logic evaluation allows the system to remain flexible, scalable, and adaptable to future needs or changing market conditions.

Beyond the technical implementation, the system also offers strategic benefits. It facilitates knowledge transfer from experienced staff to newer team members and supports more transparent, traceable decision-making. However, successful adoption will depend heavily on user acceptance and seamless integration into existing workflows. Continued development and usability improvements will be crucial to ensure that the tool is not only functional but also embraced by its intended users.

In conclusion, the proposed system represents a practical and forward-looking step toward more data-driven, standardized, and intelligent pricing in industrial sales. Its modular design and reliance on well-established industrial standards position it as a viable foundation for future enhancements and broader deployment.

ACKNOWLEDGMENTS

This work was supported by the ITEA3 programme, within the EUREKA Cluster ITEA, and funded by the German Federal Ministry of Education and Research (BMBF) under grant number FKZ 01IS21084 (InnoSale).

REFERENCES

1. Dock Team (2025) "Sales Engineering Guide: How to close complex sales" [online]. Available at: https://www.dock.us/library/sales-engineering-guide (Accessed: 14 June 2025).
2. ifak-prototypes (2025a) "FCLE - Fuzzy Control Language Engine" [online]. *GitHub repository*, Available at https://github.com/ifak-prototypes/fuzzy_control_language_engine (Accessed: 30 July 2025).
3. ifak-prototypes (2025b) "BMIProvider" [online], *GitHub repository*, Available at https://github.com/ifak-prototypes/bmi_provider (Accessed: 30 July 2025).
4. InnoSale Consortium (2025), "Project Web-Site" [online]. Available at: https://www.innosale.eu/ (Accessed: 13 June 2025).
5. InnoSale Project (2025), "Deliverable D3.5: Optimal Pricing Component". EUREKA ITEA cluster. Available at: https://itea4.org/index.php/project/workpackage/deliverable/document/download/467/InnoSale_D35_OptimalPricingComponent_final.pdf (Accessed: 17 July 2025).
6. International Electrotechnical Commission (2000) "Programmable controllers – Part 7: Fuzzy control programming", *IEC Standard No. 61131-7:2000* [online]. Available at: https://webstore.iec.ch/en/publication/4556 (Accessed: 31 July 2025).
7. Lark Developers (2024) "lark-parser" version 1.2.2 [software]. Available at: https://github.com/lark-parser/lark (Accessed: 11 July 2025).
8. Zadeh, L.A. (1965) "Fuzzy sets", Information and Control, 8(3), pp. 338–353. doi: 10.1016/S0019-9958(65)90241-X.